



MAISim Deployment

Installation, setup and the use of MAISim - Mobile Agent Malware Simulator

Rafał Leszczyna, Marcelo Masera, Igor Nai Fovino



EUR 23506 EN - 2008

The Institute for the Protection and Security of the Citizen provides research-based, systems-oriented support to EU policies so as to protect the citizen against economic and technological risk. The Institute maintains and develops its expertise and networks in information, communication, space and engineering technologies in support of its mission. The strong cross-fertilisation between its nuclear and non-nuclear activities strengthens the expertise it can bring to the benefit of customers in both domains.

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information

Address: Rafal Leszczyna TP 210; Via Enrico Fermi 2749; 21027 Ispra (VA); ITALY
E-mail: rafal.leszczyna@jrc.it
Tel.: +39 0332 786715
Fax: +39 0332 789576

<http://ipsc.jrc.ec.europa.eu/>
<http://www.jrc.ec.europa.eu/>

Legal Notice

Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of this publication.

***Europe Direct is a service to help you find answers
to your questions about the European Union***

**Freephone number (*):
00 800 6 7 8 9 10 11**

(*) Certain mobile telephone operators do not allow access to 00 800 numbers or these calls may be billed.

A great deal of additional information on the European Union is available on the Internet.
It can be accessed through the Europa server <http://europa.eu/>

JRC 47167

EUR 23506 EN
ISSN 1018-5593

Luxembourg: Office for Official Publications of the European Communities

© European Communities, 2008

Reproduction is authorised provided the source is acknowledged

Printed in Luxembourg

1. Introduction

This report describes the deployment issues related to *MAISim - Mobile Agent Malware Simulator* - a mobile agent framework which aims at simulation of *malware* – malicious software that run on a computer and make the system behaving in a way wanted by an attacker [1].

MAISim was introduced in our previous report [2] in which we described its composition and functions, and provided the details of the simulation environment in which MAISim is deployed and the auxiliary parts which support the experiments performed with MAISim. We also presented the case study where MAISim was applied to the verification of the security of an existent fully operative combined cycle electric power plant.

In this report we are providing more technical details related to the installation and use of the framework.

Our Action: *Security of Critical Networked Infrastructures (SCNI)* aims at facilitating the description, assessment and governance from the security point of view of critical networked infrastructures¹, including information systems, communication networks, electricity and other energy networks and water networks. The main interest is in cross-border and European-wide issues.

The action concentrates on the cybersecurity and topological aspects of infrastructures and their interdependencies, and studies their vulnerabilities (at the technological and system levels), the potential malicious threats that might affect them, the related detrimental attacks, and the countermeasures that can be put in place for securing those systems. It also studies the conditions and potential means for making decisions on security matters, estimating the impact of these decision, and facilitating the interaction among the stakeholders.

The focus is on providing policy makers and the stakeholders of critical infrastructures with information and instruments for a better understanding of the risks, for the qualitative and quantitative evaluation of the security issues, for the determination of the security condition of systems. From the technological perspective, the action studies the security of industrial control systems (e.g. SCADA, protection and defence systems, monitoring systems), of communication infrastructures (e.g. Internet protocols and WAN), and their application in concrete industrial environments (e.g. electric power).

One of our studies concentrates on developing a systematic approach for the identification and assessment of security risk threats to information systems. The approach is based on the systematic planning, performance and description of experiments with simulations of attacks affecting control and supervision systems. We analyse the network of a critical infrastructure and on

¹ Critical Infrastructures are defined as organisations or facilities of key importance to public interest whose failure or impairment could result in detrimental supply shortages, substantial disturbance to public order or similar dramatic impact [3]. Today most of critical infrastructures depend highly on the underlying communication networks.

the basis of our observations we reconstruct it in our laboratory. In this configuration we implement attack scenarios. Then analyse results in order to evaluate impact of the attack, test robustness and identify countermeasures. The description, preparation, execution and results of the experiments will constitute the information source for trust cases i.e. documented bodies of evidence that provide demonstrable and valid arguments that a critical infrastructure is adequately safe and secure.

The development of MAISim resulted from the lack of software and methodology for simulation of malware, while malware attacks are the most frequent in the Internet and they pose a serious threat against critical networked infrastructures.

The paper is organised as follows: in Section 2 we recall (with a slight update) from the previous report the brief overview of the related works. In Section 3 we shortly explain what MAISim is and what its main aims are. MAISim was developed using the Mobile Agents technology. Section 4 provides a short overview of it. In Section 5 we describe JADE – the agent platform used for the development of MAISim. The components of the framework and the environment in which it was deployed during our experiments are presented in Sections 6 and 7. Finally, Sections 8-12 bring in the information on the deployment, setup and use of MAISim.

2. Related Work

We haven't been able to identify any compound frameworks for performing simulations of diverse types of malware. However there are documented studies on simulation of particular malware families such as computer viruses and worms.

The studies on virus simulation tools span between:

- *Educational simulators* i.e. programs demonstrating the effect of virus infection [4]. This group of programs include Virus Simulation Suite written in 1990 by Joe Hirst, which is a collection of executables that 'simulate the visual and aural effects of some of the PC viruses' [5]. Another example is Virlab [6] from 1993, which simulates the spread of DOS computer viruses, and provides a course on virus prevention. (As it can be noticed, the programs are quite out of date, and today they would rather serve just as a historical reference.)
- *Anti-virus testing simulators* i.e. programs which are supposed to simulate viral activity, in order to test anti-virus programs without having to use real, potentially dangerous, viruses. Unfortunately, it seems that only one solution of this type was developed, namely Rosenthal Virus Simulator [7]. The simulator is a set of programs which provide 'safe and sterile, controlled test suites of sample virus programs', developed for 'evaluating anti-virus security measures without harm or contamination of the system' [7]. Again the applicability of the suite is limited since it was written ten years ago.

Concerning the simulation of worms, the prevalent work was done on developing mathematical models of worm propagation [8][9][10][11], which base on epidemiological equations that describe spread of real-world diseases. The empirical approaches concentrated mainly on single-node worm spread simulators [12][13][14][15], which are dedicated to run on one machine. Only few distributed worm simulations were implemented [16][17][18][19]. However, in all of these approaches, also the network, over which the simulated worm spreads, is simulated. Still there is a need for a simulation tool allowing simulations of malware in an arbitrary, real, physical network of computers.

Also Trojan Simulator [20] has limited applicability in our studies. It was developed for evaluating effectiveness of anti-Trojan software, and as such fulfils its purpose. However from the point of view of our experiments, it lacks the behavioural part, since the Trojan malicious activities (e.g. stealthy task execution which consumes processor time or sending packets over network) are not simulated.

3. MAISim Framework

MAISim – (Mobile Agent Malware Simulator) Framework is a software toolkit which aims at simulation of malicious software in computer network of an arbitrary information system. The framework aims at reflecting the behaviours of various families of malware (worms, viruses, malicious mobile code etc.) and various species of malware belonging to the same family (e.g. macro viruses, metamorphic and polymorphic viruses etc.). The simulated software can refer to well-known malware (e.g. Code Red, Nimda, SQL Slammer) but also it can simulate generic behaviours (file sharing propagation, e-mail propagation) and non-existent configurations (which supports the experiments aiming at predicting the system behaviour in the face of new malware).

MAISim Framework was developed using the technology of mobile agents.

4. Mobile Agents

Mobile agents are the *software agents* able to roam network freely, to spontaneously relocate themselves from one device to another.

Software agents are software components that are [21]:

- *Autonomous* – able to exercise control over their own actions.
- *Proactive* (or *goal-oriented* or *purposeful*) – goal oriented and able to accomplish goals without prompting from a user, and reacting to changes in an environment.
- *Social* (or *socially able* or *communicative*) – able to communicate both with humans and other agents.

Software agents operate on *agent platforms*. *Agent platform* is an execution environment for agents, which supplies the agents with various functionalities characteristic for the agent paradigm (such as agent intercommunication, agent autonomy, yellow pages, mobility etc.).

Agent platforms are deployed horizontally over multiple hardware devices through containers. On each device at least one container may be set up. Each container is an instance of a virtual machine and it forms a virtual agent network node. Containers make agent platform independent from underlying operating systems. Mobile agents are able to migrate from one container to another. Consequently, when containers are deployed on different devices, mobile agents can migrate between different devices.

Agent platforms can be imagined as agent communities where agents are managed and are given the means to interact (communicate and exchange services). Many agent communities may coexist at the same time. Depending on the implementation of the platform, agents may be able to leave one community (platform) and join another².

Mobile Agent approach was chosen for the development of MAISim because it particularly fits this purpose. Agents have much in common with malicious programs. Similarly to worms and viruses, they have the ability of relocating themselves from one computer to another. They are also autonomous as the worms are. At the same time they operate on agent platform which forms a type of sandbox facilitating their control.

5. JADE

MAISim is dedicated for the JADE (Java Agent DEvelopment Framework) agent platform.

JADE is a fully Java based agent platform which complies with the FIPA³ specifications. It is provided by means of:

- Software framework which facilitates the implementation of multi-agent systems through a middleware which supports agent execution and offers various additional features (such as a Yellow Pages service or support for agents' mobility).
- Set of graphical tools that support the debugging and deployment phases.

JADE is licensed under Lesser General Public License (LGPL), meaning that users can unlimitedly use both binaries and code of the platform. During over seven years of its development JADE has become very popular among the members of agent community and now it is probably the most often used agent platform. JADE is continuously developed, improved and maintained, not only by the developers from the Telecom Italia Lab (Tilab), where it was originated, but also by contributing JADE community members [31][32].

² Further information on software agents an interested reader can find in [23][24][25][26][27][28][29][30][30].

³ www.fipa.org

Further details on the choice of JADE for our works can be found in [33].

6. MAISim Components

MAISim Toolkit provides:

- Multiple classes of MAISim agent (extensions of JADE `Agent` class).
- Various behavioural patterns implemented as agent behaviours⁴ (extensions of `Behaviour` class).
- Diverse migration/replication patterns implemented as agent behaviours (extensions of `Behaviour` class).

The MAISim agent class is the basic agent code which implements the standard agent functionalities related to its management on the agent platform, its communication skills and the characteristics related to the nature of simulated malicious software. This code will be propagated across the attacked machines.

To render it operative, the code must be extended with instances of the behaviour classes and the migration/replication patterns. Depending on the chosen behaviour(s) and the migration/replication patterns, the instances of the same agent class will be created on the attacked host, or instances of another agent class from the toolkit.

The behavioural patterns comprise definitions of agent behaviours aiming at imitating malicious activities of malware (such as scanning for vulnerabilities of operating system, sending and receiving packets, verifying if certain conditions are met etc.) but *without their harmful influence on the system*. They are implemented in Java as extensions of the `Behaviour` class provided by JADE framework. The patterns include operations such as disabling network adapter, enabling a local firewall to operate in all-block mode or starting a highly processor time consuming task etc. They facilitate showing detrimental effects of malware activities but in contrary to their prototypes they are fully controlled. They demonstrate, for example, that after malware infection, it is no longer possible to connect to the host, or that the host's performance is affected etc. To support the demonstrative aspect of experiments also some patterns with audio-visual effects were developed. For example, to facilitate the observation of malware diffusion in the network, a sound can be played by the agent after it arrived to a new container⁵.

Migration and replication patterns describe the ways in which MAISim agent migrates across the attacked hosts. The patterns implement malware propagation models as well as user-configured propagation schemas. The latter allow defining such characteristics as: which subnetworks of the evaluated system will be affected, in which order, at what relative time etc.

⁴ In agents terminology the agent's *behaviour* is a set of actions performed in order to achieve the goal. It represents a task that an agent can perform [34].

⁵ Interesting studies on using sound for network monitoring are described in [35].

A particular choice of one of MAISim agent classes, extended with a chosen behavioural and migration/replication patterns is called a *malware template* - i.e. a template of malicious software. In other words, a malware template indicates a selection and configuration of Java classes (MAISim agent, one or more behavioural patterns and one or more migration/replication patterns) selected from MAISim Toolkit in order to simulate a particular instance of malware.

7. Simulation Environment

The simulations of attacks are performed in the simulation environment whose main part - *Mirrored Information System* - aims at reconstructing the information system of the evaluated infrastructure. This part is flexibly configured depending on the particular needs. For example, for the infrastructure of a power plant we mirror the process network (interconnecting diverse subsystems of the energy production process), the field network (interconnecting controllers and field devices), the corporate network etc (see Figure 1).

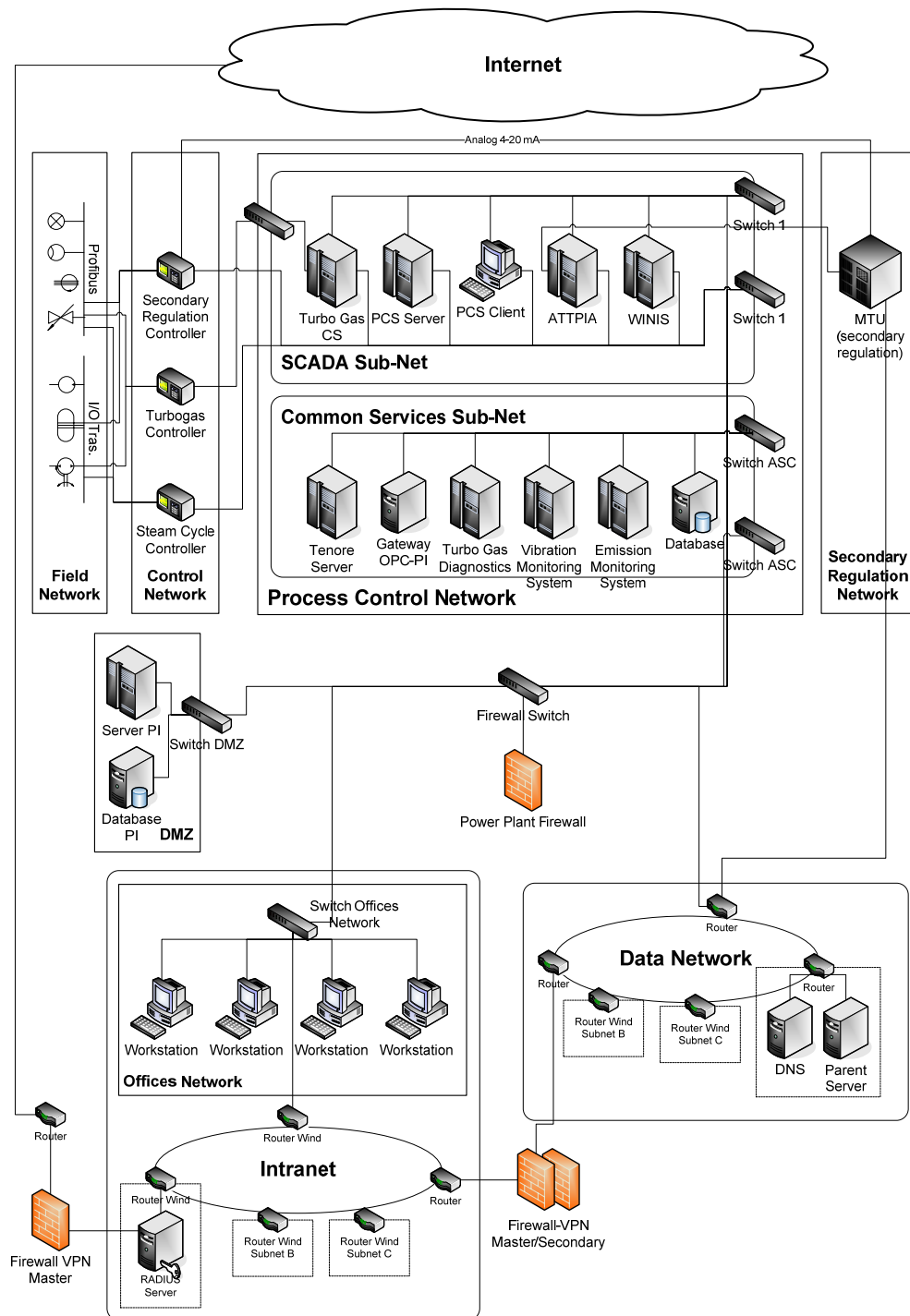


Figure 1 Reconstructed information system of a power plant.

Additionally the environment comprises the auxiliary parts which support the configuration, performance and observation of the experiments or provide any other auxiliary functionality:

- *Threat and Attack Simulator*, which aims at providing conditions for reconstructing attacks and threats that can jeopardise the analysed information system. This is the part of the simulation environment where the simulated attacks are configured and launched. Since there are various and diverse attacks, when designing this part of the simulation environment, we pay attention to assuring high flexibility and easiness of configuration. The Threat and Attack Simulator allows managing virtual subnetworks and creating multiple virtual network nodes. These, together with the hosts, are easily configurable and provided with diverse resources. Particularly they include various software i.e. operating systems and the specialised programs for developing attacker tools and for performing the attacks.
- *Observer Terminal*, which allows monitoring the traffic of the Mirrored Information System in order to evaluate the effects caused by the simulated attacks on the system. It tracks all the malicious or anomalous events happening in the Mirrored Information System during the tests and experiments, and stores them in the central database.
- *Vulnerabilities and Countermeasures Repository*, where we store all information about system vulnerabilities and the relative countermeasures. It is composed of two sub-systems: the Vulnerabilities and Countermeasures Database and the Binaries Repository. In the former we store knowledge about existing and known vulnerabilities, threats, attacks and countermeasures, while the latter is devoted to storing and cataloguing attack tools, such as packet generators, Trojan horses and root-kits, and other executable code to be used in security experiments carried out in the simulation environment. The Vulnerabilities and Countermeasures Repository is implemented within the InSAW framework [36][37].
- *Testbed Master Administrator*, used to remotely manage both the network and the experiments. It manages the operations related to the initiation and termination of experiments and allows real time observation of the behaviour of each system during simulations.
- *Horizontal Services*, responsible for providing services that are needed for the efficient management of the simulation environment such as backup services or file sharing services.

Further details about the simulation environment can be found in [1].

8. MAISim Deployment and Setup

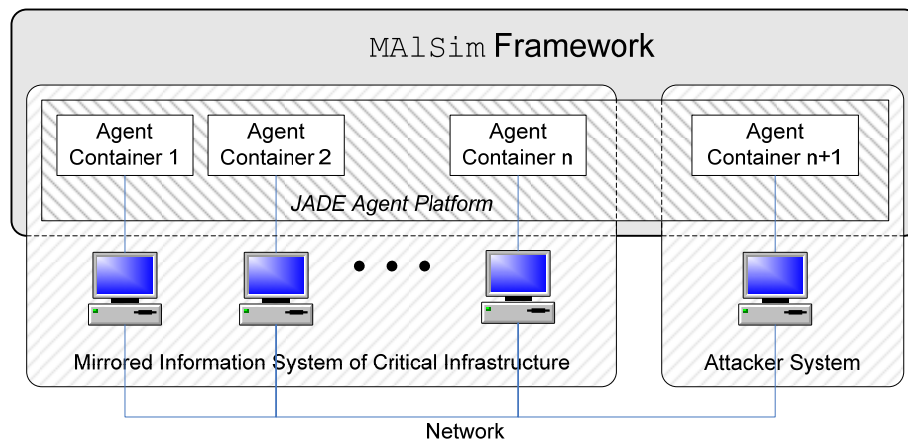


Figure 2: MAISim deployment.

As shown in Figure 2 JADE has to be deployed over all hosts participating in the experiments with MAISim. In our configuration these are the computers of Mirrored Information System and of the Attack and Threat Simulator. Java-based JADE is flexibly installable on various operating systems, and we deploy it on various distributions of Linux (Debian, Ubuntu, CentOS) and Microsoft Windows.

MAISim setup comprises the following steps:

1. An attack scenario should be taken from the repository. An attack scenario is a sequence of steps taken during attack. It describes the whole 'script' of an attack, written for all participants (the attacker, the victims, the third parties).
2. According to the chosen scenario an appropriate malware template should be selected from the repository and configured. If none of existing templates fits the attack scenario, a new MAISim template is developed.
3. Creating a live instance of malware template involves extending a MAISim agent with a migration schema (through adding agent behaviours from the repository) and a malicious behaviour.

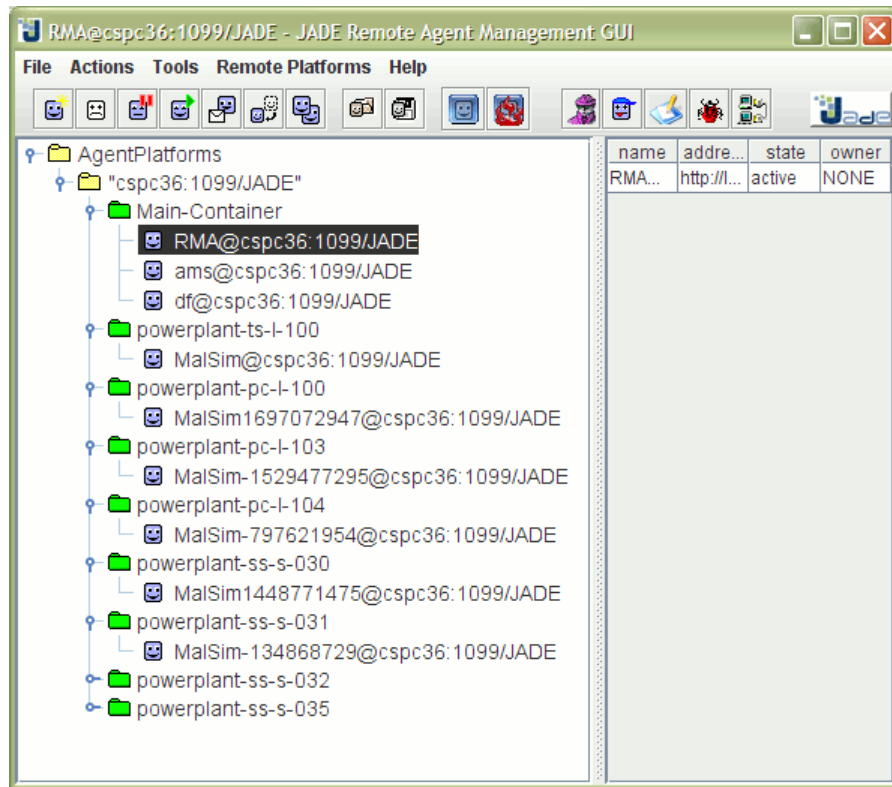


Figure 3: MAISim Framework takes advantage of JADE GUI for control and observation of experiments.

The experiments are controlled through the graphical interface of the JADE main container installed on a PC of Attack and Threat Simulator. As shown in Figure 3 the graphical console allows also observation of the diffusion of the simulated malware.

9. Windows Installation

To deploy MAISim on Widows platforms the following steps should be followed:

1. *Java installation.* We recommend to download and to set up Java Platform, Standard Edition (Java SE) from <http://java.sun.com/>.
2. *JADE download.* JADE is available for download at <http://jade.tilab.com/>. At the first visit to the site a user is required to register to it. The registration is free. In the download section, after accepting the terms of licence, users are provided with the choice of ZIP files containing elements of JADE environment. We recommend downloading the full bundle which comprises all JADE components.

3. *JADE unpacking.* We recommend unzipping the contents of the JADE installation ZIP file to Program Files. As a result the folder `c:\Program Files\jade\` with its subfolders will be created.
4. *Setting system variables.* In the Control Panel, the System icon should be double-clicked. When the "System Properties" window appears, the "Advanced" tab should be selected. There the "Environment Variables" button should be pressed. The CLASSPATH variable to system variables should be added⁶ and assigned with the following value⁷:

```
.;%jadePath%\jade.jar;%jadePath%\jadeTools.jar;%jadePath%\iiop.jar;%jadePath%\http.jar
```

A user can either substitute %jadePath% in the string with the path where JADE is installed, for example `c:\Program Files\jade` or he/she can add another variable – jadePath with the path to JADE given as a value.

10. Linux Installation

When deploying MAISim on Linux platforms a user follows the following steps:

1. *Java installation.* We recommend to download and to set up Java Platform, Standard Edition (Java SE) from <http://java.sun.com/>.
2. *JADE download.* JADE is available for download at <http://jade.tilab.com/>. At the first visit to the site a user is required to register to it. The registration is free. In the download section, after accepting the terms of licence, users are provided with the choice of ZIP files containing elements of JADE environment. We recommend downloading the full bundle which comprises all JADE components.
3. *JADE unpacking.* The ZIP file(s) of JADE should be unzipped to one of user directories, for example to `/usr/local/lib/jade/`.
4. *Setting system variables.* The following lines should be added⁸ to `.bashrc`⁹ (in the user's /home directory):

```
JAVA_HOME=".: /usr/java/jdkX.X.X_XX/ "
```

⁶ Or modify the current if it already exists

⁷ Attention should be paid to the primary dot character of the string.

⁸ Attention should be paid to the primary dot character of the string.

⁹ Assuming that Java was installed to `/usr/java/jdkX.X.X_XX/` directory and that JADE was unzipped to `/usr/local/lib/jade/`.

```
export JAVA_HOME
```

```
PATH=$PATH:/usr/java/jdkX.X.X_XX/bin  
export PATH
```

```
CLASSPATH=".:usr/local/lib/jade/lib/jade.jar:usr/local/lib/jade/lib/http.jar:usr/local/lib/jade/lib/iiop.jar:usr/local/lib/jade/lib/jadeTools.jar"  
export CLASSPATH
```

11. Setting up MAISim

Setting up MAISim environment includes:

1. *Launching the main container of JADE.* A user applies the following command in the command line:

```
java jade.Boot -gui
```

¹⁰)

2. *Adding new containers (new PCs to the environment).*

```
Java      jade.Boot      -container      -container-name  
<container-name> -host <host-IP-or-name>
```

where:

<container-name> - the name of the new container which will be added to the JADE platform. Other words – the name by which the new PC will be represented in the JADE platform (and on in the interface).

<host-IP-or-name> - the IP (or the name) of the host where the main container of JADE was launched

For example:

```
java jade.Boot -container -container-name pc-l-100 -  
host 139.166.10.11
```

¹⁰ Using the `classpath` argument can be handy e.g.

```
java -classpath  
"%CLASSPATH%":usr/local/lib/j.jar:usr/local/lib/jade/lib/http  
.jar:usr/local/lib/jade/lib/iiop.jar:usr/local/lib/jade/lib/j  
adeTools.jar jade.Boot -container -container-name pc-l-100 -  
host 139.166.10.11
```

12. Beginning with the Experiments

After the whole environment is set up, a user can begin with the experiments. Usually the experiments are started from the main container, where an appropriate MAISim agent is launched. This is done by clicking on the icon with the round smiling face which symbolises an agent, or by choosing action "Start new agent" from menu. When the window for the new agent appears, user provides the name of the Java class of the appropriate MAISim agent, for example - `MalwareSimAgent3` and the name of the agent. The former can be arbitrary, for example – `malsim3`.

References

- [1] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, USA, November 2003.
- [2] Leszczyna R., Fovino I. N., Masera M.. MAISim - Mobile Agent Malware Simulator. JRC Scientific and Technical Report, EUR 23026 EN. Luxembourg: Office for Official Publications of the European Communities, 2007.
- [3] Federal Office for Information Security (BSI). BSI annual report 2003. Internet, 2003. Available at <http://www.bsi.bund.de/english/publications/annualreport/index.htm> (last access: October 30, 2007).
- [4] Sarah Gordon. Are good virus simulators still a bad idea? *Network Security*, 1996(9):7–13, September 1996.
- [5] Joe Hirst. Virus simulation suite. Internet, 1990.
- [6] Thomas Faistenhammer, Martin Klöck, Karlhorst Klotz, Thomas Krüger, Peter Reinisch, and Jenny Wagner. Virlab 2.1. Internet, October 1993. Available at <http://kklotz.de/html/virlab.html> (last access: October 29, 2007).
- [7] Rosenthal Engineering. Rosenthal virus simulator. Internet, 1997.
- [8] Monirul I. Sharif, George F. Riley, and Wenke Lee. Comparative study between analytical models and packet-level worm simulations. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 88–98, Washington, DC, USA, 2005. IEEE Computer Society.
- [9] Symantec Research Labs. Symantec worm simulator. Internet, 2005.
- [10] Dan Ellis. Worm anatomy and model. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malware*, pages 42–50, New York, NY, USA, 2003. ACM.
- [11] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malware*, pages 51–60, New York, NY, USA, 2003. ACM.
- [12] Michael Liljenstam, Yougu Yuan, BJ Premore, and David Nicol. A mixed abstraction level simulation model of large-scale internet worm

- infestations. In *MASCOTS '02: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*, page 109, Washington, DC, USA, 2002. IEEE Computer Society.
- [13] Michael Liljenstam, David M. Nicol, Vincent H. Berk, and Robert S. Gray. Simulating realistic network worm traffic for worm warning system design and testing. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malware*, pages 24–33, 2003.
 - [14] Arno Wagner, Thomas Dübendorfer, Bernhard Plattner, and Roman Hiestand. Experiences with worm propagation simulations. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malware*, pages 34–41, New York, NY, USA, 2003. ACM.
 - [15] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 3, pages 1901–1910, April 2003.
 - [16] Kalyan S. Perumalla and Srikanth Sundaragopalan. High-fidelity modeling of computer network worms. *acsac*, 00:126–135, 2004.
 - [17] Songjie Wei, Jelena Mirkovic, and Martin Swamy. Distributed worm simulation with a realistic internet model. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 71–79, Washington, DC, USA, 2005. IEEE Computer Society.
 - [18] Songjie Wei and Jelena Mirkovic. A realistic simulation of internet-scale events. In *Valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodologies and tools*, page 28, New York, NY, USA, 2006. ACM Press.
 - [19] Filiol Éric, Franc, E., Gubbioli, A., Moquet, B., & Roblot, G. Combinatorial optimisation of worm propagation on an unknown network. *International Journal in Computer Science*, 2 (2), pages 124 – 131, 2007. Available at vx.netlux.org (last access: March 7, 2008).
 - [20] Mischel Internet Security. Trojan simulator. Internet, 2003. Available at <http://www.misec.net/trojansimulator/> (last access: October 29, 2007).
 - [21] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *JADE -A White Paper*, September 2003.

- [22] David Chess, Colin Harrison, and Aaron Kershenbaum. Mobile agents: Are they a good idea? Technical Report RC 19887 (December 21, 1994 Declassified March 16, 1995), IBM Research, Yorktown Heights, New York, 1994. Available at <http://citeseer.ist.psu.edu/chess95mobile.html>.
- [23] Davis Chess, Benjamin Grosz, Colin Harrison, David Levine, Colin Parris, and Gene Tsudik. Itinerant agents for mobile computing. *IEEE Personal Communications*, 2(5):34–49, 1995. Available at <http://citeseer.ist.psu.edu/article/chess95itinerant.html>.
- [24] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96)*, volume 1193, Berlin, Germany, 1996. Springer-Verlag New York, Inc. Available at <http://citeseer.ist.psu.edu/franklin96is.html>.
- [25] Antonio Carzaniga, Gian Pietro Picco, and Giovanni Vigna. Designing distributed applications with a mobile code paradigm. In *Proceedings of the 19th International Conference on Software Engineering*, Boston, MA, USA, 1997. Available at <http://citeseer.ist.psu.edu/carzaniga97designing.html>.
- [26] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, 1998. Available at <http://citeseer.ist.psu.edu/fuggetta98understanding.html>.
- [27] Dejan S. Milojevic. Trend wars: Mobile agent applications. *IEEE Concurrency*, 7(3):80–90, 1999. Available at <http://dlib.computer.org/pd/books/pd1999/pdf/p3080.pdf>.
- [28] Bennet S. Yee. A sanctuary for mobile agents. In *Proceedings of the DARPA Workshop on Foundations for Secure Mobile Code*, Monterey, USA, March 1997. Available at <http://citeseer.ist.psu.edu/article/yee97sanctuary.html> (last access: May 08, 2006).
- [29] Robert S. Gray, David Kotz, George Cybenko, and Daniela Rus. Mobile agents: Motivations and state-of-the-art systems. Technical Report TR2000-365, Dartmouth College, Hanover, NH, 2000. Available at <http://citeseer.ist.psu.edu/gray00mobile.html>.
- [30] W. Jansen and T. Karygiannis. Nist special publication 800-19 -mobile agent security, 2000. Available at <http://citeseer.ist.psu.edu/jansen00nist.html>.

- [31] Telecom Italia Lab. Java Agent DEvelopment Framework. Website.
<http://jade.tilab.com/>.
- [32] Giovanni Caire. *JADE tutorial: application-defined content languages and ontologies*, June 2002.
- [33] Rafal Leszczyna. Evaluation of agent platforms. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Ispra, Italy, June 2004.
- [34] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *Jade programmers guide*, February 2003.
- [35] Gilfix, M., and Couch, A. L. Peep (the network auralizer): Monitoring your network with sound. In *Lisa '00: Proceedings of the 14th USENIX conference on system administration*. Pages 109-118. Berkeley, CA, USA: USENIX Association, 2000.
- [36] Fovino I. N., Masera M., and Decian A., Integration of Cyber-Attack within Fault Trees. In: *17th European Safety And Reliability Conference (ESREL)*, Vol. 3, Pp. 2571-2578, June 2007.
- [37] Masera M. and Fovino I. N., A Service Oriented Approach to the Assessment of Infrastructure Security, Vol. 253 of *IFIP International Federation for Information Processing*, Pp. 367 - 380. Springer, Eric Goetz and Sujeet Shenoj Ed., 2008.

European Commission

EUR 23506 EN – Joint Research Centre – Institute for the Protection and Security of the Citizen

Title: MAISim Deployment

Author(s): Rafał Leszczyna, Marcelo Masera, Igor Nai Fovino

Luxembourg: Office for Official Publications of the European Communities

2008 – 16 pp. – 21.0 x 29.7 cm

EUR – Scientific and Technical Research series – ISSN 1018-5593

Abstract

This report describes the deployment issues related to *MAISim - Mobile Agent Malware Simulator* - a mobile agent framework which aims at simulation of *malware* – malicious software that run on a computer and make the system behaving in a way wanted by an attacker. MAISim was introduced in our previous report where we described its composition and functions, and provided the details of the simulation environment in which MAISim is deployed and the auxiliary parts which support the experiments performed with MAISim. In this report we are providing more technical details related to the installation and use of the framework.

How to obtain EU publications

Our priced publications are available from EU Bookshop (<http://bookshop.europa.eu>), where you can place an order with the sales agent of your choice.

The Publications Office has a worldwide network of sales agents. You can obtain their contact details by sending a fax to (352) 29 29-42758.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

